Why Relation DBMS became Object-Relational?

• Goal: Illustrate type-mismatch (SQL2, Google Maps)
  • a.k.a, semantic gap, impedance mismatch
• Maps have Geometric Objects
  • Reference Frame, e.g., Cartesian Coordinates
  • Points, Edges, Rectangles, Polygons
Entity-Relationship Diagram for Simple Geometry

• Geometric Objects
  • Reference Frame, e.g., Cartesian Coordinates
  • Points, Edges, Rectangles, Polygons

• Entity Relationship Diagram

Rectangles

Edges

Points
3NF Table Design for Simple Geometry

• Geometric Objects
  • Reference Frame, e.g., Cartesian Coordinates
  • Points, Edges, Rectangles, Polygons

• Entity Relationship Diagram

• 3NF Tables
  • Point (\text{Pid}, x, y)
  • Edge (\text{Eid}, \text{Length})
  • Rectangle (\text{Rid}, \text{Rname})
  • Starts\_or\_Ends (\text{Eid}, \text{Pid})
  • Boundary (\text{Rid}, \text{Eid})
3NF Table decompose Unit-Square Across many Tables!

- Geometric Objects
  - Reference Frame, e.g., Cartesian Coordinates
  - Points, Edges, Rectangles, Polygons

- Entity Relationship Diagram

- 3NF Tables
  - Point (\( P_{id}, x, y \))
  - Edge (\( E_{id}, \text{Length} \))
  - Rectangle (\( R_{id}, R_{name} \))
  - Starts_or_Ends (\( E_{id}, P_{id} \))
  - Boundary (\( R_{id}, E_{id} \))

- Example
  - Unit Square

<table>
<thead>
<tr>
<th>Pid</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Eid</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>1</td>
</tr>
<tr>
<td>E2</td>
<td>1</td>
</tr>
<tr>
<td>E3</td>
<td>1</td>
</tr>
<tr>
<td>E4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rid</th>
<th>Rname</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>UnitSq</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Eid</th>
<th>Pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>P1</td>
</tr>
<tr>
<td>E1</td>
<td>P2</td>
</tr>
<tr>
<td>E2</td>
<td>P2</td>
</tr>
<tr>
<td>E2</td>
<td>P3</td>
</tr>
<tr>
<td>E3</td>
<td>P3</td>
</tr>
<tr>
<td>E4</td>
<td>P4</td>
</tr>
<tr>
<td>E4</td>
<td>P1</td>
</tr>
</tbody>
</table>
Exercise

• Write SQL2 queries for the following:
  • Q1: List all rectangles with origin as a *corner* point.
  • Q2: List all rectangles containing origin as an *inside* point.

• Given 3NF Tables
  • Point (Pid, x, y)
  • Edge (Eid, Length)
  • Rectangle (Rid, Rname)
  • Starts_or_Ends (Eid, Pid)
  • Boundary (Rid, Eid)
SQL Expression for Corner Points

• Q1: List all rectangles with origin as a corner point.

```
SELECT Rid
FROM <natural join of Point, Edge, Rectangle, Starts_or_Ends, Boundary>
WHERE (x = 0) AND (y = 0)
```

• Relatively straight forward SQL for this course,
  • However, joining 5 tables is non-trivial for average programmer
  • In addition, it is costly to join so many tables to retrieve a simple object!
SQL Expression for **Inside Points**

- Q2 : List all rectangles containing origin as an inside point.

  ```sql
  SELECT Rid, MIN(x), MIN(y), MAX(x), MAX(y)
  FROM <natural join of Point, Edge, Rectangle, Starts_or_Ends, Boundary>
  GROUP BY Rid
  HAVING (0 BETWEEN MIN(x) AND MAX(x) )
     AND  (0 BETWEEN MIN(y) AND MAX(y) )
  ```

- Q2 is not straight-forward with for many of us!
- Imagine further complexity going from rectangles to polygons !
- Performance penalty of joining 5 table
Why Relational DBMS became Object-Relational DBMS?

• Object oriented thinking
  • Q1 and Q2 are much simpler in Java, Python and other Object-Oriented languages!
  • Keep pieces of unit-square together in one place instead of assembling it repeatedly!
  • Thus, Relational DBMS started adding support for objects in early 1990s.